Q1.

Source Code :

#include <iostream.h>

class bank

{

public:

int bank\_id;

char bank\_name[100], brance\_name[100];

void create\_bank();

void show\_bank() const;

};

void bank::create\_bank()

{

system("CLS");

cout << "\n\t\t\tEnter the Bank No. : ";

cin >> bank\_id;

cout << "\n\n\t\t\tEnter the Name of the Bank : ";

cin >> bank\_name;

cout << "\n\t\t\tEnter Brance Name of the Bank : ";

cin >> brance\_name;

cout << "\n\n\t\t\tBank Created.." << endl;

}

void bank::show\_bank() const

{

cout << "\n\t\t\tThe Bank No. : " << bank\_id;

cout << "\n\t\t\tName of The Bank is : ";

cout << bank\_name;

cout << "\n\t\t\tThe Bank Brance Name is : ";

cout << brance\_name << endl;

}

class account

{

public:

int ac\_no;

inbalance = 0;

int wd;

int deposit;

char name[20];

char type;

void create\_account();

void deposite();

void withdraw\_amount();

void display();

void current();

};

void account ::current()

{

if (type == 'c' || 'C')

{

display();

}

}

void account ::create\_account()

{

cout << "\n\t\t\tEnter the Name of Account Holder : ";

cin >> name;

cout << "\n\t\t\tEnter the Account No : ";

cin >> ac\_no;

cout << "\n\t\t\tEnter the Account Type (S/C) : ";

cin >> type;

cout << "\n\t\t\tEnter the Balance of Account : ";

cin >> balance;

cout << "\n\t\t Account Created...." << endl;

}

void account ::deposite()

{

cout << "\n\t\t\tDeposite The Amount in the Bank ";

cin >> deposit;

balance = balance + deposit;

}

void account ::withdraw\_amount()

{

cout << "\n\t\t\tEnter the Amount you Want to WithDraw: ";

cin >> wd;

balance = balance - wd;

}

void account ::display()

{

cout << "\n\t\tAccount No is " << ac\_no << endl;

cout << "\n\t\tAcccount Holder Name is " << name << endl;

cout << "\n\t\tThe Type of Account is " << type << endl;

cout << "\n\t\tThe Account Balance is " << balance << endl;

cout << "----------------------------------------------------" << endl;

}

class transaction : public bank, public account

{

public:

void get()

{

create\_bank();

create\_account();

}

void put()

{

deposite();

}

void out()

{

withdraw\_amount();

}

void displayt()

{

show\_bank();

display();

}

};

int main()

{

transaction obj[100];

int ch, n;

system("cls");

cout << "Enter the Number of Bank Account You Want To Create ";

cin >> n;

for (int i = 0; i < n; i++)

{

do

{

cout << "\n\n\t\t\t\t======================\n";

cout << "\t\t\t\tBANK MANAGEMENT SYSTEM";

cout << "\n\t\t\t\t======================\n";

cout << "\t\t\t\t ::MAIN MENU::\n";

cout << "\n\t\t\t\t1. NEW ACCOUNT";

cout << "\n\t\t\t\t2. DEPOSIT AMOUNT";

cout << "\n\t\t\t\t3. WITHDRAW AMOUNT";

cout << "\n\t\t\t\t4. Display";

cout << "\n\t\t\t\t5. EXIT";

cout << "\n\n\t\t\t\tSelect Your Option (1-5): ";

cin >> ch;

switch (ch)

{

case 1:

obj[i].get();

break;

case 2:

obj[i].put();

break;

case 3:

obj[i].out();

break;

case 4:

obj[i].displayt();

break;

case 5:

break;

default:

cout << "Select Correct Option" << endl;

break;

}

} while (ch != 5);

cout << "---------------------------------------" << endl;

}

system("cls");

cout << "\nThe Current Account Holder name is are as Follow " << endl;

for (int i = 0; i < n; i++)

{

obj[i].current();

}

return 0;

}

Output :

Q2.

Source Code :

#include <iostream.h>

#include <string.h>

class Word

{

public:

char str[20];

public:

void get\_string()

{

cout << "\n Enter String : ";

cin >> str;

}

void display()

{

cout << str;

}

Word operator+(Word x) // Concatenating String

{

Word s;

strcat(str, x.str);

strcpy(s.str, str);

return s;

}

int operator==(Word &t); // Comparing String

};

int Word::operator==(Word &t)

{

for (int i = 0; str[i] != '\_'; i++)

{

for (int j = 0; t.str[j] != '\_'; j++)

{

if (str[i] == t.str[j])

{

return 0;

}

else

{

return 1;

}

}

}

}

int main()

{

Word str1, str2, str3;

int result = 0;

system("cls");

str1.get\_string();

str2.get\_string();

cout << "\n ----------------------------------------------";

cout << "\n\n First String is : ";

str1.display(); // Displaying First String

cout << "\n\n Second String is : ";

str2.display(); // Displaying Second String

cout << "\n ----------------------------------------------";

str3 = str1 + str2; // String is concatenated. Overloaded '+' operator

cout << "\n\n Concatenated String is : ";

str3.display();

result = str1 == str2; // Comparing two strings. Overloaded '==' operator

if (result == 0)

{

cout << "\n\n Both Strings are Equal";

}

else

{

cout << "\n\n Both Strings are not Equal";

}

return 0;

}

Q3.

Source Code :

/\* C++ Program display Student Marksheet using Multiple inheritance \*/

#include <iostream.h>

class student

{

int roll;

char name[25], add[25],city[25];

public:

student()

{

cout << " Welcome In The Student Information System " << endl;

}

void getdata()

{

cout << "\n Enter The Student Roll No : ";

cin >> roll;

cout << "\n Enter The Student Name : ";

cin >> name;

cout << "\n enter ther student address ";

cin >> add;

cout << "\n enter the student city ";

cin >> city;

}

void putdata()

{

cout << "\n the student roll no: " << roll;

cout << "\n the student name: " << name;

cout << "\n the student city: " << city;

}

};

class marks : public student

{

int sub1, sub2, sub3;

double per;

public:

void input()

{

getdata();

cout << "\n enter the marks of 1st Subject : ";

cin >> sub1;

cout << "\n enter the marks of 2nd Subject : ";

cin >> sub2;

cout << "\n enter the marks of 3rd Subject : ";

cin >> sub3;

}

void output()

{

putdata();

cout << "\n Subject 1 marks : " << sub1;

cout << "\n Subject 2 marks : " << sub2;

cout << "\n Subject 3 marks : " << sub3 << "\n";

}

void calculate()

{

per = (sub1 + sub2 + sub3) / 3;

cout << "\n Total Percentage of Three Subject :: " << per << "%\n";

}

};

class percentage : public marks

{

public:

void out()

{

calculate();

}

};

int main()

{

system("cls");

percentage m1;

int ch,count = 0;

do {

cout << "\n1.Input data";

cout << "\n2.Output data";

cout << "\n3.Calculate percentage";

cout << "\n4.Exit\n";

cout << "\nEnter the choice :: ";

cin >> ch;

switch (ch)

{

case 1:

m1.input();

count++;

break;

case 2:

m1.output();

break;

case 3:

m1.out();

break;

}

} while (ch != 4);

}

Output:

Q4.

Source Code:

#include <iostream>

#include <bits/stdc++.h>

#include <conio.h>

#include <iomanip> //input-output manipulator

#include <stdlib.h>

#include <string.h>

#include <stdio.h>

#include <fstream>

#include <windows.h>

class book

{

char bno[6],bname[50], aname[20];

public:

void createbook()

{

cout << "\nNEW BOOK ENTRY...\n";

cout << "\nENTER BOOK NO.";

cin >> bno;

cout << "\nENTER BOOK NAME";

cin >> bname; // enables enter with space

cout << "\nENTER AUTHOR NAME";

cin >> aname;

cout << "\n\n\nBook Created..";

}

void showbook()

{

cout << "\nBook Number: " << bno;

cout << "\nBook Name: ";

puts(bname);

cout << "\nBook Author Name: ";

puts(aname);

}

void modifybook()

{

cout << "\nBook Number: " << bno;

cout << "\nModify Book Name :";

cin >> bname;

cout << "\nModify Author's Name :";

cin >> aname;

}

char \*retbno() // string return

{

return bno;

}

void report()

{

cout << bno << setw(30) << bname << setw(30) << aname << endl;

}

}; // class ends here

class student

{

char admno[6]; // admission no.

char name[20];

char stbno[6]; // student book no

int token; // total book of student

public:

void createstudent()

{

system("cls");

cout << "\nNEW STUDENT ENTRY...\n";

cout << "\nEnter The Admission No. ";

cin >> admno;

cout << "Enter The Student Name ";

cin >> name;

token = 0;

stbno[0] = '\0';

cout << "\n\nStudent Record Created...";

}

void showstudent()

{

cout << "\nAdmission Number : " << admno;

cout << "\nStudent Name : " << name << endl;

cout << "\nNo of Book Issued : " << token;

if (token == 1)

{

cout << "\nBook Number " << stbno;

}

}

void modifystudent()

{

cout << "\nAdmission No. " << admno;

cout << "\nModify Student Name : ";

cin >> name;

}

char \*retadmno()

{

return admno;

}

char \*retstbno()

{

return stbno;

}

int rettoken()

{

return token;

}

void addtoken()

{

token = 1;

}

void resettoken()

{

token = 0;

}

void getstbno(char t[])

{

strcpy(stbno, t);

}

void report()

{

cout << "\t" << admno << setw(20) << name << setw(10) << token << endl;

}

}; // class ends here

fstream fp, fp1; // object

book bk; // book class object

student st; // student class object

void writebook()

{

char ch;

fp.open("book\\C:\\LIB.txt", ios::out | ios::app); // write and append data

do {

system("cls");

bk.createbook();

fp.write((char \*)&bk, sizeof(book)); // size of class

cout << "\n\nDo you want to add more record...(y/n?) ";

cin >> ch;

} while (ch == 'y' || ch == 'Y');

fp.close();

}

void writestudent()

{

char ch;

fp.open("student\\C:\\LIB.txt", ios::out | ios::app); // write and append data

do {

system("cls");

st.createstudent();

fp.write((char \*)&st, sizeof(student)); // size of class

cout << "\n\nDo you want to add more record...(y/n?) ";

cin >> ch;

} while (ch == 'y' || ch == 'Y');

fp.close();

}

void displayspb(char n[])

{

cout << "\nBOOK DETAILS\n";

int flag = 0; // book not found

fp.open("book\\C:\\LIB.txt", ios::in); // read data

while (fp.read((char \*)&bk, sizeof(book)))

{

if (strcmp(bk.retbno(), n) == 0) // not case sensitive

{

bk.showbook();

flag = 1;

}

}

fp.close();

if (flag == 0) // book not found

{

cout << "\n\nBook does not exist";

}

getch();

}

void displaysps(char n[])

{

cout << "\nSTUDENT DETAILS\n";

int flag = 0; // student not found

fp.open("student\\C:\\LIB.txt", ios::in); // read data

while (fp.read((char \*)&st, sizeof(student)))

{

if (strcmp(st.retadmno(), n) == 0) // not case sensitive

{

st.showstudent();

flag = 1;

}

}

fp.close();

if (flag == 0) // student not found

{

cout << "\n\nStudent does not exist";

}

getch();

}

void modifybook()

{

char n[6];

int found = 0; // seach book of given data

system("cls");

cout << "\n\nMODIFY BOOK RECORD...";

cout << "\n\nEnter the book no. ";

cin >> n;

fp.open("book\\C:\\LIB.txt", ios::in | ios::out);

while (fp.read((char \*)&bk, sizeof(book)) && found == 0)

{

if (strcmp(bk.retbno(), n) == 0)

{

bk.showbook();

cout << "\nEnter the new details book";

bk.modifybook();

int pos = -1 \* sizeof(bk);

fp.seekp(pos, ios::cur); // back from current position

fp.write((char \*)&bk, sizeof(book));

cout << "\n\nRecord Updated";

found = 1;

}

}

fp.close();

if (found == 0)

{

cout << "\n\nRecord Not Found";

}

getch(); // press key to get out

}

void modifystudent()

{

char n[6];

int found = 0; // seach book of given data

system("cls");

cout << "\n\nMODIFY STUDENT RECORD...";

cout << "\n\nEnter the Admission no. ";

cin >> n;

fp.open("student\\C:\\LIB.txt", ios::in | ios::out);

while (fp.read((char \*)&st, sizeof(student)) && found == 0)

{

if (strcmp(st.retadmno(), n) == 0)

{

st.showstudent();

cout << "\nEnter the new details of student";

st.modifystudent();

int pos = -1 \* sizeof(st);

fp.seekp(pos, ios::cur); // back from current position

fp.write((char \*)&st, sizeof(student));

cout << "\n\nRecord Updated";

found = 1;

}

}

fp.close();

if (found == 0)

{

cout << "\n\nRecord Not Found";

}

getch(); // press key to get out

}

void deletestudent()

{

char n[6];

int flag = 0;

system("cls");

cout << "\n\n\n\tDELETE STUDENT...";

cout << "\n\nEnter the Admission no> : ";

cin >> n;

fp.open("student\\C:\\LIB.txt", ios::in | ios::out);

fstream fp2;

fp2.open("temp\\C:\\LIB.txt", ios::out);

fp.seekg(0, ios::beg);

while (fp.read((char \*)&st, sizeof(student)))

{

if (strcmp(st.retadmno(), n) != 0)

{

fp2.write((char \*)&st, sizeof(student));

}

else

{

flag = 1; // student found

}

}

fp2.close();

fp.close();

remove("student\\C:\\LIB.txt");

rename("temp\\C:\\LIB.txt", "student\\C:\\LIB.txt"); // data after deletion moved to temp

if (flag == 1)

{

cout << "\n\n\tRecord Deleted..";

}

else

{

cout << "\n\nRecord not Found";

}

getch();

}

void deletebook()

{

char n[6]; // book no.

int flag = 0;

system("cls");

cout << "\n\n\n\tDELETE BOOK...";

cout << "\n\nEnter the Book no> : ";

cin >> n;

fp.open("book\\C:\\LIB.txt", ios::in | ios::out);

fstream fp2; // New onject

fp2.open("Temp\\C:\\LIB.txt", ios::out); // temp having data else than that to be deleted

fp.seekg(0, ios::beg);

while (fp.read((char \*)&bk, sizeof(book)))

{

if (strcmp(bk.retbno(), n) != 0)

{

fp2.write((char \*)&st, sizeof(book));

}

else

{

flag = 1; // student found

}

}

fp2.close();

fp.close();

remove("book\\C:\\LIB.txt");

rename("Temp\\C:\\LIB.txt", "book\\C:\\LIB.txt"); // data after deletion moved to temp

if (flag == 1)

{

cout << "\n\n\tRecord Deleted... ";

}

else

{

cout << "\n\nRecord not Found";

}

getch();

}

void displayalls()

{

system("cls");

fp.open("student\\C:\\LIB.txt", ios::in); // read mode

if (!fp)

{

cout << "File Could Not Be Open";

getch();

return; // press any key and return

}

cout << "\n\n\t\tStudent List\n\n";

cout << "==================================================================\n";

cout << "\tAdmission No." << setw(10) << "Name" << setw(20) << "Book Issued\n";

cout << "==================================================================\n";

while (fp.read((char \*)&st, sizeof(student)))

{

st.report();

}

fp.close();

getch();

}

void displayallb()

{

system("cls");

fp.open("book\\C:\\LIB.txt", ios::in); // read mode

if (!fp)

{

cout << "File Could Not Be Open";

getch();

return; // press any key and return

}

cout << "\n\n\t\tBook List\n\n";

cout << "==================================================================\n";

cout << "\tBook No." << setw(20) << "Book Name" << setw(25) << "Book Author\n";

cout << "==================================================================\n";

while (fp.read((char \*)&bk, sizeof(book)))

{

bk.report();

}

fp.close();

getch();

}

void bookissue()

{

char sn[6], bn[6];

int found = 0, flag = 0;

system("cls");

cout << "\n\nBOOK ISSUE...";

cout << "\n\n\tEnter Admission no.";

cin >> sn;

fp.open("student\\C:\\LIB.txt", ios::in | ios::out);

fp1.open("book\\C:\\LIB.txt", ios::in | ios::out);

while (fp.read((char \*)&st, sizeof(student)) && found == 0)

{

if (strcmp(st.retadmno(), sn) == 0) // compare admsn no.

{

found = 1;

if (st.rettoken() == 0) // if book not issued

{

cout << "\n\n\tEnter The Book No.";

cin >> bn;

while (fp1.read((char \*)&bk, sizeof(book)) && flag == 0)

{

if (strcmp(bk.retbno(), bn) == 0) // compare book no.

{

flag = 1;

st.addtoken();

st.getstbno(bk.retbno()); // pass book no.

int pos = -1 \* sizeof(st);

fp.seekg(pos, ios::cur);

fp.write((char \*)&st, sizeof(student));

cout << "\n\n\tBook Issued Successfully\n\n Please Note The Book Issue Date On Backside Of Your Book And Return Book Within 15 Days, Otherwise Fine Of 15 Rs Per Day";

}

}

if (flag == 0)

{

cout << "Book No. Does Not Exists";

}

}

else

{

cout << "You Have Not Returned The Last Book";

}

}

}

if (found == 0)

{

cout << "Student Record Not Exists...";

}

getch();

fp.close();

fp1.close();

}

void bookdeposit()

{

char sn[6], bn[6];

int found = 0, flag = 0, day, fine;

system("cls");

cout << "\n\nBOOK DEPOSIT...";

cout << "\n\n\tEnter Admission no. Of Student";

cin >> sn;

fp.open("student\\C:\\LIB.txt", ios::in | ios::out);

fp1.open("book\\C:\\LIB.txt", ios::in | ios::out);

while (fp.read((char \*)&st, sizeof(student)) && found == 0)

{

if (strcmp(st.retadmno(), sn) == 0) // compare admsn no.

{

found = 1;

if (st.rettoken() == 1) // if book issued

{

while (fp1.read((char \*)&bk, sizeof(book)) && flag == 0)

{

if (strcmp(bk.retbno(), st.retstbno()) == 0)

{

flag = 1;

bk.showbook();

cout << "\n\n Book Deposited In No. Of Days";

cin >> day;

if (day > 15)

{

fine = (day - 15) \* 1;

cout << "\n\n Fine = " << fine;

}

st.resettoken();

int pos = -1 \* sizeof(st);

fp.seekg(pos, ios::cur);

fp.write((char \*)&st, sizeof(student));

cout << "\n\n\tBook Deposited Successfully";

}

}

if (flag == 0)

{

cout << "Book No. Does Not Exists";

}

}

else

{

cout << "No Book Issued";

}

}

}

if (found == 0)

{

cout << "Student Record Not Exists...";

}

getch();

fp.close();

fp1.close();

}

void start()

{

system("cls");

cout << "LIBRARY ";

cout << "MANAGEMENT ";

cout << "SYSTEM";

cout << " \n by: Rishit";

getch();

}

void adminmenu()

{

system("cls");

int ch2;

cout << "\n\n\n\tADMINISTRATOR MENU";

cout << "\n\n\n\t1.CREATE STUDENT RECORD";

cout << "\n\n\n\t2.DISPLAY ALL STUDENT RECORD";

cout << "\n\n\n\t3.DISPLAY SPECIFIC STUDENT RECORD";

cout << "\n\n\n\t4.MODIFY STUDENT RECORD";

cout << "\n\n\n\t5.DELETE STUDENT RECORD";

cout << "\n\n\n\t6.CREATE BOOK";

cout << "\n\n\n\t7.CREATE ALL BOOKS";

cout << "\n\n\n\t8.DISPLAY SPECIFIC BOOK";

cout << "\n\n\n\t9.MODIFY BOOK RECORD";

cout << "\n\n\n\t10.DELETE BOOK RECORD";

cout << "\n\n\n\t11.BACK TO MAIN MENU";

cout << "\n\n\n\tPLEASE ENTER YOUR CHOICE(1-11)";

cin >> ch2;

switch (ch2)

{

case 1:

writestudent();

break;

case 2:

displayalls();

break;

case 3:

char num[6];

system("cls");

cout << "\n\n\t Please enter admission no.";

cin >> num;

displaysps(num);

break;

case 4:

modifystudent();

break;

case 5:

deletestudent();

break;

case 6:

writebook();

break;

case 7:

displayallb();

break;

case 8:

{

char num[6];

system("cls");

cout << "\n\n\tPlease enter book no.";

cin >> num;

displayspb(num);

break;

}

case 9:

modifybook();

break;

case 10:

deletebook();

break;

case 11:

return;

default:

cout << "Invalid choice";

}

adminmenu();

}

int main()

{

char ch;

system("cls");

start();

do

{

system("cls");

cout << "\n\n\n\t MAIN MENU";

cout << "\n\n\n\t1 BOOK ISSUE";

cout << "\n\n\n\t2 BOOK DEPOSIT";

cout << "\n\n\n\t3 ADMINISTRATOR MENU";

cout << "\n\n\n\t4 EXIT";

cout << "\n\n\n\t PLEASE SELECT YOUR OPTION(1-4)";

ch = getche();

switch (ch)

{

case '1':

bookissue();

break;

case '2':

bookdeposit();

break;

case '3':

adminmenu();

break;

case '4':

exit(0);

break;

default:

cout << "INVALID CHOICE";

}

} while (ch != 4);

return 0;

}

Output:

Q5.

Source Code:

#include <iostream.h>

class event

{

private:

string name;

public:

// constructor initializes the event

event()

{

cout << "Enter the Event Name ";

cin >> name;

}

void getevent()

{

cout << "The Event Name is " << name << endl;

}

};

int main()

{

// dynamically declare event object

event \*ptr = new event();

// call getevent() function

ptr->getevent();

// ptr memory is released

delete ptr;

return 0;

}

Output:

Q6.

Source Code:

#include <iostream.h>

class structure

{

protected:

int stackarray[4], top;

structure()

{

top = -1;

}

~structure()

{

"Stack destructed!\n";

}

};

class data\_structure : public structure

{

public:

void push(int n)

{

if (top == 3)

cout << "Stack overflow\n";

else

{

top++;

stackarray[top] = n;

}

}

void pop()

{

if (top == -1)

cout << "Stack underflow\n";

else

{

cout << stackarray[top] << " popped successfully\n";

top--;

}

}

void display()

{

if (top == -1)

cout << "Stack empty\n";

else

{

for (int i = 0; i <= top; i++)

cout << "The Value is : " << i + 1 << ": " << stackarray[i] << endl;

}

}

};

int main()

{

system("cls");

data\_structure s1;

char ch;

do {

cout << "Press...\n"<< "1 to push\n"<< "2 to pop\n"<< "3 to display\n"<< "4 to exit\n\n";

cin >> ch;

switch (ch) {

case '1':

int n;

cout << "Enter the value : \n";

cin >> n;

s1.push(n);

break;

case '2':

s1.pop();

break;

case '3':

s1.display();

break;

}

} while (ch != '4');

return 0;

}

Output:

Q7.

Source Code:

#include <bits/stdc++.h>

bool isOperator(char c)

{

return (!isalpha(c) && !isdigit(c));

}

int getPriority(char C)

{

if (C == '-' || C == '+')

return 1;

else if (C == '\*' || C == '/')

return 2;

else if (C == '^')

return 3;

return 0;

}

string infixToPostfix(string infix)

{

infix = '(' + infix + ')';

int l = infix.size();

stack<char> char\_stack;

string output;

for (int i = 0; i < l; i++)

{

// If the scanned character is an

// operand, add it to output.

if (isalpha(infix[i]) || isdigit(infix[i]))

output += infix[i];

// If the scanned character is an

// ‘(‘, push it to the stack.

else if (infix[i] == '(')

char\_stack.push('(');

// If the scanned character is an

// ‘)’, pop and output from the stack

// until an ‘(‘ is encountered.

else if (infix[i] == ')')

{

while (char\_stack.top() != '(')

{

output += char\_stack.top();

char\_stack.pop();

}

// Remove '(' from the stack

char\_stack.pop();

}

// Operator found

else

{

if (isOperator(char\_stack.top()))

{

if (infix[i] == '^')

{

while (getPriority(infix[i]) <= getPriority(char\_stack.top()))

{

output += char\_stack.top();

char\_stack.pop();

}

}

else

{

while (getPriority(infix[i]) < getPriority(char\_stack.top()))

{

output += char\_stack.top();

char\_stack.pop();

}

}

// Push current Operator on stack

char\_stack.push(infix[i]);

}

}

}

while (!char\_stack.empty())

{

output += char\_stack.top();

char\_stack.pop();

}

return output;

}

string infixToPrefix(string infix)

{

/\* Reverse String

\* Replace ( with ) and vice versa

\* Get Postfix

\* Reverse Postfix \* \*/

int l = infix.size();

// Reverse infix

reverse(infix.begin(), infix.end());

// Replace ( with ) and vice versa

for (int i = 0; i < l; i++)

{

if (infix[i] == '(')

{

infix[i] = ')';

}

else if (infix[i] == ')')

{

infix[i] = '(';

}

}

string prefix = infixToPostfix(infix);

// Reverse postfix

reverse(prefix.begin(), prefix.end());

return prefix;

}

// Driver code

int main()

{

string s;

cout << "Enter the string : ";

cin >> s;

cout << infixToPrefix(s) << std::endl;

return 0;

}

Output:

Q8.

Source Code:

#include <iostream.h>

#include <cstdlib>

// Node Declaration

struct node

{

int info;

node \*next;

node \*prev;

} \* head, \*tail;

// Class Declaration

class dqueue

{

public:

int top1, top2;

void insert();

void del();

void display();

dqueue()

{

top1 = 0;

top2 = 0;

head = NULL;

tail = NULL;

}

};

// Main: Contains Menu

int main()

{

int choice;

dqueue dl;

while (1)

{

cout << "\n-------------" << endl;

cout << "Operations on Deque" << endl;

cout << "\n-------------" << endl;

cout << "1.Insert Element into the Deque" << endl;

cout << "2.Delete Element from the Deque" << endl;

cout << "3.Traverse the Deque" << endl;

cout << "4.Quit" << endl;

cout << "Enter your Choice: ";

cin >> choice;

cout << endl;

switch (choice)

{

case 1:

dl.insert();

break;

case 2:

dl.del();

break;

case 3:

dl.display();

break;

case 4:

exit(1);

break;

default:

cout << "Wrong Choice" << endl;

}

}

return 0;

}

// Insert Element in Doubly Ended Queue

void dqueue::insert()

{

struct node \*temp;

int ch, value;

if (top1 + top2 >= 50)

{

cout << "Dequeue Overflow" << endl;

return;

}

if (top1 + top2 == 0)

{

cout << "Enter the value to be inserted: ";

cin >> value;

head = new (struct node);

head->info = value;

head->next = NULL;

head->prev = NULL;

tail = head;

top1++;

cout << "Element Inserted into empty deque" << endl;

}

else

{

while (1)

{

cout << endl;

cout << "1.Insert Element at first" << endl;

cout << "2.Insert Element at last" << endl;

cout << "3.Exit" << endl;

cout << endl;

cout << "Enter Your Choice: ";

cin >> ch;

cout << endl;

switch (ch) {

case 1:

cout << "Enter the value to be inserted: ";

cin >> value;

temp = new (struct node);

temp->info = value;

temp->next = head;

temp->prev = NULL;

head->prev = temp;

head = temp;

top1++;

break;

case 2:

cout << "Enter the value to be inserted: ";

cin >> value;

temp = new (struct node);

temp->info = value;

temp->next = NULL;

temp->prev = tail;

tail->next = temp;

tail = temp;

top2++;

break;

case 3:

return;

break;

default:

cout << "Wrong Choice" << endl;

}

}

}

}

// Delete Element in Doubly Ended Queue

void dqueue::del()

{

if (top1 + top2 <= 0)

{

cout << "Deque Underflow" << endl;

return;

}

int ch;

while (1)

{

cout << endl;

cout << "1.Delete Element at first" << endl;

cout << "2.Delete Element at last" << endl;

cout << "3.Exit" << endl;

cout << endl;

cout << "Enter Your Choice: ";

cin >> ch;

cout << endl;

switch (ch)

{

case 1:

head = head->next;

head->prev = NULL;

top1--;

break;

case 2:

tail = tail->prev;

tail->next = NULL;

top2--;

break;

case 3:

return;

break;

default:

cout << "Wrong Choice" << endl;

}

}

}

// Display Doubly Ended Queue

void dqueue::display()

{

struct node \*temp;

int ch;

if (top1 + top2 <= 0)

{

cout << "Deque Underflow" << endl;

return;

}

while (1)

{

cout << endl;

cout << "1.Display Deque from Beginning" << endl;

cout << "2.Display Deque from End" << endl;

cout << "3.Exit" << endl;

cout << endl;

cout << "Enter Your Choice: ";

cin >> ch;

cout << endl;

switch (ch)

{

case 1:

temp = head;

cout << "Deque from Beginning:" << endl;

while (temp != NULL)

{

cout << temp->info << " ";

temp = temp->next;

}

cout << endl;

break;

case 2:

cout << "Deque from End:" << endl;

temp = tail;

while (temp != NULL)

{

cout << temp->info << " ";

temp = temp->prev;

}

temp = tail;

cout << endl;

break;

case 3:

return;

break;

default:

cout << "Wrong Choice" << endl;

}

}

}

Output: